

Remarks/Argument

Please acknowledge receipt of the formal drawings filed on Sep. 6, 2001.

In reply to the indication of allowable subject matter in paragraph 2 of the Official Action, claims 9 and 28 have been re-written in independent form including all of the limitations of their respective base claims 8 and 27, there being no intervening claims. Claim 10 is dependent upon claim 9, and claim 29 is dependent on claim 28. In addition, claims 21 and 50 includes limitations similar to those found in claims 9 and 28. Claim 28 has also been amended to correct an obvious typographical error.

Claims 25 and 40 have been amended to explicitly recite the comparison referred to in the original claims 25 and 40. (See also original claim 19).

In paragraph 3 on page 2 of the Official Action, claims 1-50 were rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-30 of U.S. Patent 6,324,581. In reply, enclosed herewith is a Terminal Disclaimer by the Assignee, EMC Corporation, which is submitted in order to obviate this rejection.

In paragraph 5 on page 4 of the Official Action, claims 1-8, 11-20, 27, and 30-49 were rejected under 35 U.S.C. 103(a) as being unpatentable over Yasue (U.S. Patent 6,289,345) in view of Schmuck et al. (U.S. Patent 6,032,216). Applicants respectfully traverse.

The invention relates generally to data storage systems, and more particularly to network file servers. (Applicants' specification, page 1, lines 2-4). In particular, data consistency problems may arise if concurrent client access to a read/write file is permitted through more than one data mover. (Applicants' specification, page 3, lines 1-2.)

The invention provides a method of operating a file server in a data network. The file server receives a request for metadata about a file to be accessed. The request is received from a data processing device in the data network. In response to the request for metadata, the file server grants to the data processing device a lock on at least a portion of the file, and returns to the data processing device metadata of the file including information specifying data storage locations in the file server for storing data of the file. (Claim 1.)

For example, as shown in applicants' FIG. 3, before reading or writing to the file system 62, a client first issues a request for metadata to the data mover 61. The data mover 61 responds by placing an appropriate lock on the file to be accessed, and returning metadata including pointers to where the data to be accessed is stored in the file system. The client uses the metadata to formulate a read or write request sent over the bypass data path to the file system 62. If the write request changes the file attributes, then the client writes the new file attributes to the data mover 61 after the data is written to the file system 62. (Applicants' specification, page 14, lines 18-25.)

In another example, shown in FIG. 4, a second client 88 accesses the first file system 83 in the fashion described above with reference to FIG. 3, and a third client 89 accesses the second file system 84 in the fashion described above with reference to FIG. 3. For example, to access the first file system 83, the second client 88 sends a metadata request to the first data mover 81. The first data mover 81 places a lock on the file to be accessed, and returns metadata including pointers to the data in the file to be accessed. The second client 88 uses the pointers to formulate a corresponding data access command sent over the bypass data path 92 to the first file system

83, and any read or write data is also communicated over the bypass data path 92 between the first file system 83 and the second client 88. (Applicants' specification, page 16, lines 7-15.)

The file server 60 in FIG. 3 provides direct data sharing between network clients 64, 65 by arbitrating and coordinating data access requests. The data mover 61 grants file lock request from the clients 64, 65 and also provides metadata to the clients 64, 65 so that the clients can access data storage 62 in the cached disk array 63 over a data path that bypasses the data mover 61. The data mover 81, 82 and the clients 88, 89 in FIG. 4 may operate in a similar fashion. (Applicant's specification, page 55, lines 18-24.)

The network file server architecture of FIG. 4 allows file sharing among heterogeneous clients, and supports multiple file access protocols concurrently. The architecture permits clients using traditional file access protocols to inter-operate with clients using the new distributed locking and metadata management protocol for direct data access at the channel speed of the data storage devices. This provides a scaleable solution for full file system functionality for coexisting large and small files. (Applicants' specification, page 58, line 30, to page 59, line 5.)

The policy of the Patent and Trademark Office has been to follow in each and every case the standard of patentability enunciated by the Supreme Court in Graham v. John Deere Co., 148 U.S.P.Q. 459 (1966). M.P.E.P. § 2141. As stated by the Supreme Court:

Under § 103, the scope and content of the prior art are to be determined; differences between the prior art and the claims at issue are to be ascertained; and the level of ordinary skill in the pertinent art resolved. Against this background, the obviousness or nonobviousness of the subject matter is determined. Such secondary considerations as commercial success, long felt but unsolved needs, failure of others, etc., might be utilized to give light to the circumstances

surrounding the origin of the subject matter sought to be patented. As indicia of obviousness or nonobviousness, these inquiries may have relevancy.

148 U.S.P.Q. at 467.

The problem that the inventor is trying to solve must be considered in determining whether or not the invention would have been obvious. The invention as a whole embraces the structure, properties and problems it solves. In re Wright, 848 F.2d 1216, 1219, 6 U.S.P.Q.2d 1959, 1961 (Fed. Cir. 1988).

Paragraph 6 on page 4 of the Official Action cites Yasue for disclosing “a method of operating a file server in a data network, said method comprising: the file server receiving a request for metadata about a file to be accessed [i.e. acquire information][Abstract], the request being received from a data processing device in the data network [(a), Figure 8; col 4, lines 16-25; and col 7, lines 44-51]; and returning to the data processing device metadata of the file including information specifying data storage location in the file server for storing data of the file [(b), Figure 8; col 2, lines 36-43; and col 7, lines 44-51.]”

However, Yasue discloses a design information management system comprised of a computer aided design application program layered over a relational database system for a database distributed over a number of bulk servers. In order to locate model data indicating the shape of an object, a metaserver is accessed to obtain an identification of a particular one of the bulk servers and an identification of a location on the bulk server onto which the model data is stored. (See Yasue claims 11 and 12, col. 11 line 30 to col. 12 line 10.)

It is not seen where Yasue discloses the claimed “information specifying data storage locations in the file server for storing data of the file ...” Yasue’s identification of a location on the bulk server onto which the model data is stored does not specify data storage locations in a file server for storing data of the file. Yasue’s specific examples of such identification of a location on the bulk server onto which the model data is stored are the location sites /USR/TEMP/1432, /USR/TEMP/1111, and /USR/TEMP/2222 in FIG. 3B, WS1:/USR/TMP/ in FIG. 6B and 7B, and the “ACCESS VOLUME” entries /USR/TMP4321, /USR/TMP/4444, and /OPT2/CAD/DATA in FIG. 12C. In other words, Yasue’s identification of a location on the bulk server is a pathname of a file. (See also Yasue col. 5, line 62, and col. 7 line 23.) A file name does not specify data storage locations in the file server for storing data of the file because a primary function of the file manager in the file server is to map file names to data storage locations allocated to the files by the file manager. (See applicants’ specification, page 2, lines 17-20.) In contrast, the applicants’ specification describes the applicants’ file server providing the user or client with “pointers to where the data to be accessed is stored in the file system.” (Page 14, lines 22-23.) This permits the client to use the pointers to “formulate a read or write request sent over the bypass data path to the file system 62.” (Page 14, lines 22-23). For example, the pointers point “to where the file data resides in the cached disk array storing the file system.” (Page 31 lines 13-14).

The Official Action (page 5) further says that Yasue does not specifically disclose “in response to the request for metadata, the file server granting to the data processing device a lock on at least a portion of the file.” The Official Action (page 5) cites Schmuck for disclosing “in response to the request for metadata, the file server granting to the data processing device a lock

on at least a portion of the file [col 33, lines 37-57].” However, column 33, lines 37-57 of Schmuck simply disclose the granting of distributed range locks so that processors may have concurrent access to a file:

Serializing accesses to different regions in a file to which processes on different nodes write in parallel is done by distributed byte range locks. When a process needs to lock a byte range, it first needs to acquire an appropriate byte range token. The byte range token represents the node's access rights to a portion of a file. Thus, if a node holds a byte range token for file X for range (100, 200) in read mode, it means that the node may safely read that portion of the file. However, to prevent stealing the token, the node must lock the token before the actual read, since if another node needs to write the same portion, it might steal the token. Locking the token prevents the steal. After the read has completed, the token is unlocked.

One can view tokens as a way of "caching" locks. When a node needs to lock a portion of a file, it needs to lock the token. At first, it will acquire a token and lock it. Once the operation is finished and the token is unlocked, it is still resident at the node. Thus, subsequent operations on the same region would not need to access the token authority. Only when the token is stolen will a new request for the token be needed.

It is not seen where the token manager of Schmuck (11 in FIG. 1) grants such a distributed range lock to a node (computer 1, 2, or 3 in FIG. 1) in response to a request for metadata. Instead, the cited passage of Schmuck suggests that a node requests a byte range token from the token manager 11 when a node needs to read or write to a file.

Paragraph 6 on page 5 of the Official Action concludes: “It would have been obvious to a person skill[ed] in the art at the time the invention was made to combine Yasue and Schmuck because Schmuck’s locking mechanism would provide efficient basic file control in a shared disk environment for multiple computers [Schnuck, col. 3, lines 50-57].” Applicants respectfully contend, however, that the claimed invention would not result simply from combining Yasue and

Schmuck, and there is no proper motivation for modifying such a combination to arrive at the claimed invention.

The design information management system of Yasue (Fig. 1) has workstations 30 that access a common metaserver 10 and a number of bulk servers 20. The common metaserver 10 stores management information, location management information, and assignment of drawing number information. The bulk servers store 3-dimensional CAD data. Apparently a workstation may access the metaserver to determine the bulk server and the name of a file containing a desired drawing, and then the workstation may access the CAD data for the desired drawing from this bulk server.

The shared disk file system in FIG. 1 of Schmuck includes a shared token manager 11 which provides locking facilities for computers which are considered nodes 1, 2, and 3 participating in the management of a file system. [Schmuck, col. 3, line 66, to col. 4, line 4.] “There is no file system server in the path for either data or metadata. Any available path can be used avoiding a server as a bottleneck or as a single point of failure. Since the central functions in the lock manager have no attachment to a specific computer, they can be migrated from computer to computer to satisfy performance and availability needs.” [Schmuck col. 4, lines 18-24.]

If a person of ordinary skill in the art were asked to combine Yasue and Schmuck, one might expect the workstations of Yasue to access a token manager of Schmuck when the workstations would read or write to the 3-dimensional CAD data files in the bulk servers to permit byte range locking of the CAD data files and achieve concurrent write access. However, this would not result in applicants’ claimed invention because the metaserver still would not

return to a workstation metadata of the file including information specifying data storage locations in the file server for storing data of the file. Nor would the request to the metaserver from a workstation result in the granting to a workstation a lock on at least a portion of the file. There is nothing in Yasue nor Schmuck to suggest that the metadata server should be taking over the conventional file manager functions of mapping file names to data storage locations allocated to the files by the file manager, or that the file locking should be done in response to the request for metadata.

Where the prior art references fail to teach a claim limitation, there must be “concrete evidence” in the record to support an obviousness rejection. “Basic knowledge” or “common sense” is insufficient. *In re Zurko*, 258 F.3d 1379, 1385-86, 59 U.S.P.Q.2d 1693, 1697 (Fed. Cir. 2001).

Moreover, the applicants’ specification, page 17, lines 1-10, teach that use of the invention is desirable only under particular circumstances:

Whenever a client has a bypass data path to a file system and can therefore send data access commands to the file system without passing through a data mover computer, the client can potentially access all of the files in the file system. In this situation, the client must be trusted to access only the data in a file over which the client has been granted a lock by the data mover that owns the file system to be accessed. Therefore, the methods of client access as described above with reference to FIGS. 2 and 3 have a security risk that may not be acceptable for clients located in relatively open regions of the data network. The method of client access as described above with reference to FIG. 3 also requires special client software, in contrast to the methods of client access as described above with reference to FIGS. 1-2 which can use standard client software.

Hindsight reconstruction, using the applicant's specification itself as a guide, is improper because it fails to consider the subject matter of the invention "as a whole" and fails to consider the invention as of the date at which the invention was made. [T]here must be some motivation, suggestion, or teaching of the desirability of making the specific combination that was made by the applicant." In re Lee, 277 F.3d 1338, 1343, 61 U.S.P.Q.2d 1430, 1435 (Fed. Cir. 2002) (quoting In re Dance, 160 F.3d 1339, 1343, 48 U.S.P.Q.2d 1635, 1637 (Fed. Cir. 1998)). "[T]eachings of references can be combined only if there is some suggestion or incentive to do so." In re Fine, 837 F.2d 1071, 1075, 5 U.S.P.Q.2d 1596, 1600 (Fed. Cir. 1988) (Emphasis in original) (quoting ACS Hosp. Sys., Inc. v. Montefiore Hosp., 732 F.2d 1572, 1577, 221 U.S.P.Q. 929, 933 (Fed. Cir. 1984)). "[P]articular findings must be made as to the reason the skilled artisan, with no knowledge of the claimed invention, would have selected these components for combination in the manner claimed." In re Kotzab, 217 F.3d 1365, 1371, 55 U.S.P.Q.2d 1313, 1317 (Fed. Cir. 2000). See, for example, Fromson v. Advance Offset Plate, Inc., 755 F.2d 1549, 1556, 225 U.S.P.Q. 26, 31 (Fed. Cir. 1985) (nothing of record plainly indicated that it would have been obvious to combine previously separate lithography steps into one process); In re Gordon et al., 733 F.2d 900, 902, 221 U.S.P.Q. 1125, 1127 (Fed. Cir. 1984) (mere fact that prior art could be modified by turning apparatus upside down does not make modification obvious unless prior art suggests desirability of modification); Ex Parte Kaiser, 194 U.S.P.Q. 47, 48 (PTO Bd. of Appeals 1975) (Examiner's failure to indicate anywhere in the record his reason for finding alteration of reference to be obvious militates against rejection).

With respect to claim 2, paragraph 7 on pages 5-6 of the Official Action cites Schmuck for disclosing a data mover computer maintaining a metadata cache in random access memory

[col. 30, lines 41-48] and the data mover computer accessing the metadata cache for obtaining the metadata that is returned to the data processing device [col. 3, lines 40-48; and col. 8, lines 58-60]. However, Schmuck [col. 30, lines 37-48] deals with a multiprocessor system in which:

In this system, a node is appointed for each file which is responsible for accessing and updating the file's metadata. This metadata node (or metanode) shares this information with other nodes upon request.

The metadata node keeps the information about the file's metadata and acts as a smart cache between the disk and all the nodes that access the file. There are situations when the metadata node (or metanode) ceases to serve this function. In order to enable smooth operation and recovery, these situation[s] need to be handled. Nodes that used to access the metanode need to elect a new metanode in a straightforward way.

However, from the cited references, it is not seen how or why metadata caching in a multi-processor system such as Schmuck (FIG. 1) should be applied to caching of metadata in a file server for providing the metadata (including the information specifying data storage locations for storing data of the file) to a data processing device (e.g., a client) in a data network. One would expect a file server caching metadata to keep to itself the information specifying data storage locations for storing data of the file. Moreover, in Schmuck (FIG. 1), the token manager (11) manages locks, and the nodes (i.e., computers 1, 2, 3) access and update the metadata.

With respect to claim 3, Schmuck col. 30, lines 55-65, further discloses that the token manager is a distributed subsystem which grants tokens to nodes. The token manager is not a file server distinct from the nodes (i.e., computers 1, 2, 3). In the applicants' claims, the file

server is distinct from the data processing device or client; in particular, the file sever receives from the data processing device a request for metadata about a file to be accessed, the file server grants the lock to the data processing device, and returns to the data processing device metadata of the file including information specifying data storage locations in the file server for storing data of the file.

With respect to claim 4, paragraph 9 on page 6 of the Official Action says: “Yasue discloses the data processing device writes data to the data storage locations in the file server [i.e. update] [col 2, lines 51-67], modifies the metadata from the file server in accordance with the data storage locations in the file server to which the data is written [col 4, lines 45-56], and sends the modified metadata to the file server [col 3, lines 11-21].” Applicants respectfully disagree. As discussed above, Yasue’s system uses files to store the model data of the objects relating to the devices and components in the assembly and component models. File names are used by the workstation programs and the metadata server to specify where the model data is stored. A file name, however, does not specify the data storage locations in the file server for storing data of the file. Therefore, the metadata in the in the metadata server of Yasue fails to specify the data storage locations in the file server for storing data of the file. Moreover, it is not seen where a workstation program in Yasue modifies the metadata in accordance with the data storage locations to which the data of the file is written. In Yuasa, the allocation and modification of the data storage locations to which the data of the file is written would be done by the bulk servers 20 and not by the workstation programs.

In a conventional file server, it is the file server that is mapping the file name to the storage locations, and modifying the metadata of the file when needed in accordance with the

data storage locations in the file server to which the data is written. In a similar fashion, in a database server, the database server would map objects in the database (such as tables and the cells in the tables) to data storage locations in the database server. Claim 4 instead defines that the data processing device is writing data to the data storage locations in the file server, and modifying the metadata from the file server in accordance with the data storage locations in the file server to which the data is written. In short, claim 4 defines that the data processing device (e.g., the client) is performing operations previously performed by the file server.

With respect to claim 6, paragraph 11 on page 7 of the Official Action acknowledge that Yasue does not disclose the elements expressly recited in claim 6, and cites Schmuck col. 42, lines 10-32 for these elements. Schmuck col. 42, lines 10-32 say:

Thus, every node eventually finds out either that it has become the new metanode or that the metanode has changed. In either case, appropriate actions are taken. If a node became a metanode, it reads the most recent metadata from disk. If a node's metanode changed, the node will re-send its own metadata updates to the new metanode since its possible that the old metanode failed before flushing these updates to disk. By using a version number for each such update, every node knows which updates are on disk and which have to be re-sent to the new metanode.

Although Schmuck discloses that a version number is associated with each metadata update, it is not seen where the data processing device includes this version identifier in a request for access to a file, so that the file server compares the version identifier from the data processing device to a version identifier of a most recent version of the metadata of the file, in order to

return the most recent version of metadata to the data processing device when the metadata of the file cached in the cache memory of the data processing device is not the most recent metadata of the file. Instead, it appears that the version number in Schmuck is used to maintain coherency in the multi-processor system of Schmuck between the metadata cache of the metanode and the metadata on disk, for example, when there is a crash of the old metanode and assignment of a new metanode.

Paragraph 11 on page 7 of the Official Action concludes: “It would have been obvious to a person skill[ed] in the art at the time that the invention was made to combine the teachings of Yasue and Schmuck because Schmuck’s teaching of versioning would allow to maintain data integrity.” Applicants respectfully disagree. Even if the meta server 10 in Yasue were provided with a metadata cache and a version identifier were associated with each metadata update in order to maintain coherency between the metadata cache and disk, there is no suggestion that the user or client (i.e., one of the workstations 30) should be provided with a metadata cache, and the user or client should include the version identifier of its cached metadata in a request for metadata, in order for the metadata server to return the most recent version of metadata to the user or client when the metadata in the cache memory of the user or client would not be the most recent metadata of the file.

Paragraph 13 on page 8 of the Official Action rejected claim 8 for similar reasons as claim 1. Applicants respectfully traverse, for the same reasons as discussed above with respect to claim 1. In addition, claim 8 further defines the operation of the client in the data processing network. The file server sends to the client metadata of the file including information specifying data storage locations in the file server for storing data of the file, and the client uses this

metadata of the file for producing at least one data access command for accessing the data storage locations in the file server, and sends the data access command to the file server to access the data storage locations of the file server. This further distinguishes Schmuck et al. because there is no differentiation of the computer nodes and disks in the multiprocessor system of FIG. 1 of Schmuck into a file server and a client in a data network.

The Official Action cites Yasue FIG. 8 and col. 9, lines 61-67. This describes a workstation in Yasue accessing the management information in the metaserver to determine the appropriate bulk server that stores model data to be requested, and requesting this model data from the appropriate bulk server. In contrast, applicants' claim 8 is directed to a client obtaining a lock on at least a portion of a file and metadata of the file in response to a request for access to a file, and using the metadata of the file to produce at least one data access command for accessing the data storage locations in the file server. Obtaining an identification of a file server storing a desired file to access and accessing the file server (as described in Yasue) should not be confused with a process of accessing a file server (as claimed in applicants' claim 8) by sending to the file server a request to access a file and in response obtaining a lock on the file and information specifying data storage locations in the file server for storing data of the file in order to produce a data access command for accessing the data storage locations in the file server .

With respect to claim 11, paragraph 14 on page 8 of the Official Action rejected claim 11 for similar reasons as stated above in claim 3. Applicants respectfully traverse, for the reasons given above with respect to claim 3.

With respect to claim 14, paragraph 17 on page 9 of the Official Action rejected claim 14 for similar reasons as stated above in claim 8. Applicants respectfully traverse, for the reasons

given above with respect to claim 8, and more importantly because claim 14 defines the particular input-output related operating system routines that are dynamically linked to the application programs of the client. It is not seen where the cited references suggest that these input-output related operating system routines of the client should intercept file access calls from client application processes to obtain from the file server locks upon at least a portion of each of the files, obtain metadata for producing data access commands for accessing data storage in the file server, produce the data access commands from the metadata, and send the data access commands to the file server in order to access the data storage of the file server. The claimed input-output routines have the effect of transferring and incorporating into the operating system level of the client various functions that were previously performed by the network file server.

The Official Action cites Figure 2 and col. 5, lines 1-11 of Yasue for disclosing dynamically linking application programs of the client with input-output related operating system routines of the client. The passage from Yasue says:

FIG. 2 is a block diagram of the present invention. Component constitution information 12, stored in the meta-database 11, indicates what other components are in a particular device and component. For example, using a tree structure, it can be seen that device A comprises components a1, a2, a3, and that component a1 comprises components a11, a12, while component a2 comprises components a21, a22. The term “device” typically refers to the top level structure in any particular diagram/drawing/figure, while the term “component” typically refers to the subparts of a device or component.

This passage of Yasue fails to suggest that input-output related operating system routines of the client should intercept file access calls from client application processes to obtain from the file server locks upon at least a portion of each of the files, obtain metadata for producing data access commands for accessing data storage in the file server, produce the data access commands from the metadata, and send the data access commands to the file server in order to access the data storage of the file server.

Paragraph 18 on page 9 of the Official Action rejected claim 15 for similar reasons as stated above with respect to claim 4. Applicants respectfully traverse, for the reasons as stated above with respect to claim 4.

Paragraph 22 on page 10 of the Official Action rejected claim 19 for similar reasons as stated above with respect to claim 6. Applicants respectfully traverse, for the reasons as stated above with respect to claim 6.

Paragraph 23 on page 10 of the Official Action rejected claim 27 for similar reasons as stated above with respect to claims 1 and 8. Applicants respectfully traverse, for the reasons as stated above with respect to claims 1 and 8.

Paragraph 25 on page 10 of the Official Action rejected claim 34 for similar reasons as stated above with respect to claim 14. Applicants respectfully traverse, for the reasons as stated above with respect to claim 14.

Paragraph 26 on page 11 of the Official Action rejected claim 35 for similar reasons as stated above with respect to claim 6. Applicants respectfully traverse, for the reasons as stated above with respect to claim 6.

Paragraph 27 on page 11 of the Official Action rejected claim 36 for similar reasons as stated above with respect to claim 1. Applicants respectfully traverse, for the reasons as stated above with respect to claim 1.

Paragraph 28 on page 11 of the Official Action rejected claim 37 for similar reasons as stated above with respect to claim 4. Applicants respectfully traverse, for the reasons as stated above with respect to claim 4.

Paragraph 29 on page 11 of the Official Action rejected claims 38 and 39 for similar reasons as stated above with respect to claim 2. Applicants respectfully traverse, for the reasons as stated above with respect to claim 2.

Paragraph 30 on page 11 of the Official Action rejected claim 40 for similar reasons as stated above with respect to claim 6. Applicants respectfully traverse, for the reasons as stated above with respect to claim 6.

Paragraph 31 on page 11 of the Official Action rejected claim 42 for similar reasons as stated above with respect to claim 8. Applicants respectfully traverse, for the reasons as stated above with respect to claim 8.

Paragraph 32 on page 11 of the Official Action rejected claims 43-45 for similar reasons as stated above with respect to claim 13-15. Applicants respectfully traverse, for the reasons as stated above with respect to claims 13-15.

Paragraph 34 on page 12 of the Official Action rejected claim 49 for similar reasons as stated above with respect to claim 6. Applicants respectfully traverse, for the reasons as stated above with respect to claim 6.

In view of the above, reconsideration is respectfully requested, and early allowance is earnestly solicited.

Respectfully submitted,

3 November 2004

date



Richard C. Auchterlonie
Reg. No. 30,607
Novak, Druce & Quigg, LLP
1000 Louisiana, Suite 5320
Houston, TX 77002
713-751-0655